

SYSTEM AND METHOD FOR PROVIDING AN APPLICATION NAVIGATOR

CLIENT MENU SIDE BAR

Background of the Invention

Cross References to Related Applications

5
AI

The following U.S. patent applications filed concurrently herewith are assigned to the same assignee hereof and contain subject matter related, in certain respects, to the subject matter of the present application, the teachings of which applications are incorporated herein by this reference:

Serial No. 09/_____, entitled "System and Method for Clustering Servers for Performance and Load Balancing", assignee docket END9-2000-0104-US1;

Serial No. 09/_____, entitled "System and Method for Front End Business Logic and Validation", assignee docket END9-2000-0105-US1;

Serial No. 09/_____, entitled "System and Method for Data Transfer With Respect to External Applications", assignee

END9 2000 0111 US1

docket END9-2000-0106-US1;

Serial No. 09/_____, entitled "System and Method for Providing a Relational Database Backend", assignee docket END9-2000-0107-US1;

5 A Serial No. 09/_____, entitled "System and Method for Providing a Role Table GUI via Company Group", assignee docket END9-2000-0108-US1;

10 Serial No. 09/_____, entitled "System and Method for Populating HTML Forms Using Relational Database Agents", assignee docket END9-2000-0109-US1;

Serial No. 09/_____, entitled "System and Method for Catalog Administration Using Supplier Provided Flat Files", assignee docket END9-2000-0110-US1.

Technical Field of the Invention

15 This invention pertains to a system and method for managing a requisition catalog on the web.

Background Art

A requisition catalog for a large enterprise in a web environment must enable very fast access to a very large database from a large number of clients. A large number of clients is required for optimal performance of a catalog system. There is a need to architect such a system so as to be scalable, that is, capable of providing that fast access to an ever increasing number of clients and a growing database or collection of databases.

10 In a requisition catalog system for a large enterprise in a web environment, there is a requirement for a system and method for insuring that all general ledger accounts associated with commodities are correct, and within this requirement for providing a description from the associated accounting system for use by customer or requester to select the correct general ledger (GL) account when doing financial validation on a requisition.

20 A requisition catalog for a large enterprise is stored in a very large database. However, a Lotus Notes database has a hard limit of 2 to 4 gigabytes of data. That is, Notes puts its entire database into one big Notes standard format(.nsf) file. Information in such a .nsf file is accessed by a system geometrically. As a result, such a system slows down significantly as it accesses larger files.

This slow down ramps up rather badly. Consequently, the hard limit is artificially put in because of this ramp up in access time. The typical solution to this problem is to split a very large database into many smaller databases.

5 Lotus Notes has a Lotus connector/Lotus script extensions connector (LCLSX). This allows connection to other databases but these script extensions are not set up to actually be the database. There is a need in the art for a system and method for utilizing Lotus script extensions in
10 combination with a relational database to provide high capacity storage without performance degradation.

In building a requisition catalog for a large enterprise with many suppliers, an automated process is needed to receive a flat file from a supplier for review by
15 a buyer before being externalized for use by requesters. While the buyer must be able to review the contents, he must be restricted from making changes to certain sensitive fields, such as changing a unit price or a unit of measure, both of which could constitute fraud. Consequently,
20 allowing the buyer to edit the flat file can't provide the level of security required. There is a need in the art to provide a buyer a means of auditing catalog content before externalizing it to production for access by requesters.

A requisition catalog for use in a web environment requires a very large database, such as an IBM DB2 database, and the functionality provided by, for example, a Lotus Notes server. However, a Lotus Notes access control list (ACL) can not be used control access to an IBM DB2 database, and the privileges on a DB2 table can be granted only by the table instance owner. Additionally, since Notes agents which access DB2 are running from a Notes server, the Notes server ID often has full access to all tables, and there is no way to limit that. That is, in a hybrid (Notes/DB2) environment, the user ID which accesses DB2 tables is the ID of the Notes server. Therefore, can't restrict access by a user to the DB2 tables. There is a need in the art for a system and method which allows certain users access to certain data in certain selected tables. That is, there is needed a system and method for providing very flexible access to DB2 tables without requiring database administrator (DBA) involvement to issue grants against the tables, and bypassing the problem caused by Notes agents all coming from the same user (the Notes server ID).

In a hybrid requisition catalog system for use in a web environment, much of the application data is stored in, for example, IBM DB2 tables. However, a web interface written in Java script and in HTML does not have functionality for

frame set in a browser. As part of security for such a browser, Netscape and Internet Explorer (IE) establish ownership of the frames: each frame is a window under control of the browser. A user can open up a session in any
5 frame desired. However, if all of the frames are not owned by same session, then these frames not commonly owned cannot see the contents of each other. In Netscape version 4.51 frame ownership was changed to protect against frame spoofing. Frame spoofing is a way other servers can use to
10 trick the owner of a frame into thinking it had created a frame actually created by someone else, enabling access to data not owned. Netscape 4.51 plugs that hole.

Consequently, in the new environment (with spoofing inhibited) a problem is presented: if a RCW application
15 needs to access an outside supplier site for information to get back through one its own the frames, as soon as data is written by that outside supplier site into one of the frames owned by RCW, ownership of the frame is transferred from RCW to the supplier; RCW no longer owns the frame and cannot
20 access the information. A system and method is needed to enable transfer of data from a supplier site to a RCW application which does not entail frame spoofing.

In a web application, such as a requisition catalog

system for a large enterprise, a fast browser interface and navigator is needed, including a very fast graphical user interface (GUI).

5 There is known in the art several Internet applications which provide a strip down, for example, the left hand side of a window that has menu of items from which a user may select. Such windows also may display a header, including header type items which will expand upon selection into a drop down list. Such windows also may include a footer including leafs which will, upon selection, change what is
10 seen on right side of screen. Such applications provide a very nice user interface for documents, with a table of contents (TOC) on the left, and on the right the selected contents.

15 However, these expandible and collapsible menus are characteristically slow. This slowness is a result of every click sending a request to a remote server, which server responds by sending information back to expand the list at the browser. Every click on the web is slow, involving
20 communications of the request and response over a relatively slow web communication link. There is, therefore, a need in the art to dramatically increase the speed of operation of a graphical user interface at a browser.

It is an object of the invention to provide a scalable database system in a web environment with optimal access performance characteristics for an expanding number of clients and a growing database.

It is an object of the invention to provide a system and method for enabling a requester to select the correct general ledger account when doing financial validation on a requisition.

It is an object of the invention to provide a system and method for utilizing Lotus script extensions in combination with a relational database to provide high capacity storage without performance degradation.

It is an object of the invention to provide a system and method for enabling a buyer a means for editing catalog content before externalizing it to production for access by requesters.

It is an object of the invention to provide a system and method for providing very flexible access to DB2 tables without requiring database administrator (DBA) involvement to issue grants against the tables, and bypassing the

problem caused by Notes agents all coming from the same user
(the Notes server ID).

It is an object of the invention to provide a system
and method utilizing an existing infrastructure including
5 Lotus Notes, Domino Go, and DB2 to combine HTML and Java
script web presentation with DB2 data.

It is an object of the invention to provide a system
and method for enabling transfer of data from a supplier
site to a RCW application which does not entail frame
10 spoofing.

It is an object of the invention to provide a system
and method for dramatically increasing the speed of
operation of a navigation frame of a GUI.

Summary of the Invention

15 A system and method for presenting a browser navigation
frame. A screen display includes a customizable side bar.
Dynamic HTML is used by a navigator responsive to user input
to change data presented at the navigation frame without
having to communicate with a remote server.

Other features and advantages of this invention will become apparent from the following detailed description of the presently preferred embodiment of the invention, taken in conjunction with the accompanying drawings.

5

Brief Description of the Drawings

Figure 1 is a high level system diagram illustrating a clustered configuration of servers for performance and load balancing.

10

Figure 2 is a diagram illustrating proxying out of URLs to clustered servers.

Figure 3 is a system diagram illustrating a specific example of a clustered configuration of servers.

Figure 4 is a diagram illustrating the frames comprising a typical screen display.

15

Figure 5 is a diagram illustrating a specific instance of the display of Figure 4.

of the userProfile class of Figure 13.

Figure 15 is a system diagram illustrating the system of Figure 1 for accessing a requisition catalog.

Figure 16 is a system diagram illustrating a system for using a staged requisition catalog built from supplier flat catalog files.

Figure 17 is a flow diagram illustrating the steps for receiving a supplier flat catalog.

Figure 18 is a flow diagram illustrating the steps executed by an application server and database server for building and accessing a requisition catalog.

Figure 19 is a diagram illustrating a user profile.

Figure 20 is a diagram illustrating a Notes agent for building an HTML page from a DB2 table.

Figure 21 is a diagram illustrating Notes agents for transferring data to an application browser session from a supplier window.

1. Clustered Servers

THE 2000s

15

09657495.090700

distributed, proxy passed, or redirected) by network
dispatcher 106 among servers S4, S5, and S6 in Domino
cluster 112. While three servers are illustrated as
configured in each of clusters 104 and 112, each cluster
5 configuration may be scaled to any number of servers.
External objects 108, which can be stored on a distributed
file system (.dfs), include graphic files, Java files, HTML
images, net.data macros, and other .nsf files external to
Domino, and in particular include configuration file proxy
10 statements 110. In this instance, external objects 108 are
stored on a .dfs and exist only once, so it is not necessary
to replicate all of the external objects to each of the
servers S4-S6. External objects 108, served in a .dfs are
graphic files, Java files, anything that would live outside
15 of the server files (also referred to .nsf files) S4-S6,
including HTML images and net.data macros. These are part
of the code implementing the Req/Cat Web application of the
preferred embodiment of the invention, but are not part of
Domino code, and are primarily for the GUI. By storing them
20 outside of cluster servers 112, performance is improved.

In order to avoid potential bottlenecks on the
clustered Domino servers 112 and in order to store a larger
amount of data than is quickly searchable in Domino, a
relational database 129, such as the IBM DB2 database, is

used to store configuration data. Data is written by the clustered servers 112 by the end user, or by batched programs stored on application server 114 that are receiving data from back-end systems 116.

5

Referring to Figure 2, in accordance with the invention, a proxy pass is used with both domino.go cluster 104 and Domino cluster 112. In accordance with a proxy pass, when a URL 120 is passed to network dispatcher 106, the NP processes that out and sprays it to any one of the configured servers. Spray means to distribute or map a URL 120 to any one of these configured servers S1-S3, S4-S6, which is the effect of mapping, as is represented by line 126, URL 120 to any of S4, S5, S6 in cluster 112. Examples of URLs include <w3.ibm.com/*>, <www.ibm.com/*>, and <w3.ibm.com/transform/reqcat/?opendatabase rccreate>.

In accordance with the invention, a unique architecture for a requisition catalog system includes a hybrid application using external objects 108 in a distributed file system off of the domino.go cluster 104 that works with network dispatcher 106 and the proxy pass capability 110 to redirect traffic to the Domino cluster 112.

These servers S4-S6 are mirror images: each has same

.nsf files. Periodically, these servers replicate back and forth so that information is contained in all of them. Data is kept consistent.

In operation, when a client comes in through browser
5 100, his request can be directed to any domino.go server S1-S3 that determines the mapping of the URL request and what type of function is needed (displaying of images or code execution on S1-S3, or connecting to an .nsf server S4-S6 to display user data).

10 .nsf servers S4-S6 then feed requests to the application server 114, which in this embodiment is a backend Req/Cat Web (RCW) server to which all data gets replicated and where the bridges and agents run. Data gets replicated out to other back-end servers (DB2, MVS, SAP) 116
15 as needed. A bridge is an application that transfers data from one server to another server. An agent is an application that runs scheduled or by request to do some processing of data.

In an exemplary embodiment, Domino.go, or virtual
20 server cluster, 104 is part of the GWA infrastructure. Any w3.ibm.com or www.ibm.com must go through a domino.go cluster 104. In accordance with the present invention a

proxy out to the Domino cluster 112 is also provided. The purpose of this is to improve scalability and performance.

Proxy statements 110 are used to ensure that the proper pieces of the application are invoked as appropriate,

5 depending upon what the end user is doing. These statements are a mapping through a configuration file of URL 120 to any clustered server 112.

Referring to Figure 3, a specific exemplary embodiment of the invention includes client browser 100 connected to
10 network dispatcher (URL redirect) 102, which is connected to GWA shared GO cluster (W3-1.IBM.COM) 104. Cluster 104 is connected to external objects including OBI 117 and DFS 118, to dedicated DB2 server 129, and to network dispatcher (proxy) 106. Dispatcher 106 is connected to dedicated
15 Domino "end user" cluster 112. Cluster 112 is connected to Blue Pages database 121 (an internal personnel database), dedicated DB2 server 129, dedicated Domino "application" server 114, and other Notes databases 119, including Skills Matching (an application for contracting technical
20 services), AMNF (an application for identifying nonmanager requisition approvers), and public address book (PAB, for user login and authentication). Application server 114 is connected to FormWave 125 (an application that does approval form routing), PRISM/Copics 123 (which are requisition

(w3.ibm.com/transform/reqcat) and Network dispatcher 106
redirects the client to the appropriate server 112, the
redirection is transparent to the client.

5 Workload is split between Domino.Go 104 and Domino 112
to improve performance.

10 File storage is provided by dynamic file system (.dfs,
or DFS) 118, which contains javascript files (.js), html,
images, and net.data macros. Domino servers 114 contain
navigation, configuration, create requisition, open
requisition, requisition invoice paid, requisition archive,
cost center, confirmations, and help databases. DB2 server
129 contains tables including: confirmations, user profiles,
zip codes, accounting data, commodity configuration data,
buyer information, routing, and catalogs.

15 An end-user front end for Req/Cat Web uses Domino
authentication to permit login to the application. The
client uses his Lotus Notes ID and pre-set http password to
"authenticate". When a client logs in at browser 100, the
Domino servers 112 (S4-S6) are configured to check the name
20 and password in public address book 119, which is a
designated server within GNA. In a further exemplary
embodiment, secure login function may be provided through

the implementation of digital certificates. Data is
retrieved from DB2 129 using LC:LSX calls via Notes agents,
or Net.Data. Net.Data is used for a catalog searching and
drill-down function. Java APIs are used for information
5 retrieval from BluePages 121.

As much processing as possible is performed
asynchronously on the back-end, application server 114.
Bridge jobs are scheduled on two levels: system level (CRON)
and notes level (Agents). Agents run periodically, say
10 hourly, to process requisitions and send them to SAP 382.
Other agents are scheduled off-peak, where ever possible.

External dependencies include FormWave for form
approval routing, BluePages for personal data for user
profile creation and approver changes, interfaces to Open
15 Buying on the Internet (OBI) server and skill matching
applications, and SAP for receiving requisitions for
purchase order (PO) creation and processing. Information
returned by SAP to Req/Cat Web includes requisition status,
PO/RFS status, PO history, negative confirmations, currency
20 codes, and configuration information.

By using frames, a large majority of preprocessing can
be performed dynamically, on the client, reducing the number

displyReqHeader() javascript function is called directly.
This form reads the information stored in reData frame and
dynamically fills the content frame with this screen.

Referring to Figures 4 and 5, as will be more fully
described hereafter, a screen display includes header frame
470, navigation frame 472, footer frame 474, temporary data
frame 476, request data frame 478, and content frame 480.
The tempData frame 476 is used as a temporary holder for
information, and to direct calls dynamically, while keeping
the current data in the screen, and making the return data
available to the application.

Table 1 shows, for the exemplary embodiment of Figure
3, the software loaded on each of the servers used for the
Req/Cat Web application.

TABLE 1 SERVERS AND SOFTWARE

| <u>Server Description</u> | <u>Software</u> |
|-------------------------------|---|
| Domino.Go Servers S1-S3 | AIX 4.3.2 Domino Go 4.6.2.6 with Denial of Service Fix Net.Data 2.0.8 DB2 CAE 5.2 DFS Client Java Runtime 1.1.6 |
| Domino End-User Servers S4-S6 | AIX 4.3.2 |

Whereas previously, customer input of ZIP code was accepted without checking. By this invention, ZIP code validation is performed at the front end by a java or SQL program call to db2 database 134. Responsive to entry of ZIP code on a requisition or to the changing of delivery information which includes ZIP code on a requisition line item, the ZIP code is validated against a database of valid ZIP codes. In an alternative embodiment, the ZIP code database is refreshed from a trusted source, and the entered or changed ZIP code is checked for valid match with respect to state and city.

A create requisition request goes to catalog search, which used to search by part number or description. Previously, this was a very limited search to just the catalogs. A search argument of %pen% was not a very crisp search for the customer. In order to improve the catalog search, in accordance with the present invention, searches may be conducted against a longer description and files up to 255 characters. Screen down searches are provided for sub-commodity. Wild card searches used to require %, but now assumes a wild card search in all cases. Searches are also provided against subcommodity. As a result, catalog searches now reference short description, long description, and catalog sub-commodity. A database catalog includes part number, short description, long description, oem part

END9 2000 0111 US1

number, commodity code. Newly added is subcommodity.

Referring to Figures 7 and 9, the method of the preferred embodiment of the invention is described for managing a chart of accounts 140. When creating a commodity
5 list, which includes expense, capital, and resale accounts 142, commodities descriptions 180 are pushed to the correct commodity group. The resulting chart of accounts 140 is available from SAP 382. Previously an administrator had input a chart of accounts. Now, a company administrator,
10 for example, may select from commodity accounts 142 the expense field, which results in drop-down display of a valid chart of accounts 144 from SAP chart of accounts 140 with account numbers 148 and commodity descriptions 146. The company administrator may then select from that valid chart
15 of accounts 144 the correct commodity to push to company/commodity document 150.

Referring to Figure 8, the process for a requester to create requisition is set forth. By way of example, a requestor creates a requisition by doing in step 154 a
20 search for "supplies", which will bring to him in step 156 a display presenting commodity W14, and thence in step 158 to a catalog (for example, a Staples catalog) which includes commodities (pens, erasers, calendars), from which the

requester can create several line items. Upon selecting
"proceed to accounting", the requester is presented a
financial summary 160 including commodity code w14 pens for
line item 1, w14 erasers for line item 2, and w14 for line
5 item 3 calendars. The user may then request display of
financial worksheet 162. In this window 162, the requester
will see a title 164 expense, which can be changed, for
example, by toggling to other categories, such as balance
sheet. Selecting G/L account 166 may drop down a list
10 showing several account codes 168 and related descriptions
170, depending upon what the company administrator has
pushed to the commodity document 150 from which financial
worksheet 162 is derived. Previously, a requester was
provided in worksheet 162 one account code choice without
15 description. By this invention, the requester is provided
correct general ledger account codes and descriptions,
resulting in less miscodes, more correct ledger entries, and
correct SAP account codes. This improved general ledger
account selection process avoids back end processing to
20 correct erroneous entries.

Thus, in accordance with a preferred embodiment of the
invention, a method is provided for creating a valid chart
of accounts from which an administrator 184 can facilitate
and enable a requisitioner to select a valid general ledger

by catalog and non-catalog, and driven based on descriptions entered by requester. A hierarchy of families may be provided as an alternative search approach. The requisitioner initiates a proceed to accounting process, which displays line items which may be selected by requisitioner; and then displays a financial worksheet created by a Java agent with fields which need to be selected or populated by the requisitioner from the company commodity document, based on purchase type, and which presents valid general ledger accounts numbers and descriptions to the requisitioner.

3. Back End

Referring to Figure 12, in a large enterprise, the requisition catalog requires a very large database. In accordance with the preferred embodiment of the invention, such a large data repository is provided by the using the IBM DB2 relational database 210. Other possible databases include Oracle, Sybase, and MSSQL.

Lotus Notes databases are built upon an object model and classes: databases, views, and documents are classes used to access Notes data. But, these classes are set to be

final and not extendible, and a Req/Cat Web database must be extendible.

Consequently, referring to Figure 12, in accordance with an embodiment of the present invention, DB2 access routines 208 are provided for accessing DB2 data 210. The Req/Cat Web application executes Lotus code, with access controlled on the code, and data obtained from and written to relational database 210.

Normally, Notes saves all data as documents. There is a save method provided for that purpose. In accordance with a preferred embodiment of the invention, the Notes save method is intercepted and stopped, and execution passed to Req/Cat Web code for saving data to DB2. In the same way, execution of a Notes open method is intercepted and stopped, and then Req/Cat Web code executed to pull information in from DB2.

Lotus Notes provides for web applications, and supports methods called webqueryopenagent, and webquerysaveagent. Notes also provides a saveoptions parameter. Setting saveoptions to zero tells Notes not to save a document.

In accordance with the preferred embodiment of the

access them. This also enables a DB2 administrator to alter a table, and only affect the one DAPI developer that wrote the specific table DAPI 202. All code is one routine, so changes to the database need only affect one piece of logic.

5 Referring to Figure 13, an example of this API implementation is illustrated. CoreDB2 220 is the core API 200 to connect to DB2 210. It contains two classes, configuration class DB2Config 222 and base class DB2Base 224.

10 Calling DB2Config 222 determines database name 230, user identifier 231, and password 232 - information that the database requires to establish a valid connection, and is passed to DAPI 208 for making that connection. (Without this method, user IDs and passwords would have to be hard
15 coded in the application.)

DB2Base 224 is extended by the DAPI 208 programmer for each DAPI 202 instance that is needed. It contains methods 240, 241 for connecting to and disconnecting from the database, a method 242 for defining the number of rows to
20 return at a time, a method 243 for getting the next group of records, methods for reads 244, inserts 245, updates 246, and deletes 247, commit 248 and rollback 249 options, and a

flag 250 to determine if all data has been retrieved.

DAPIUserProfile 226 is a class for retrieving or updating information about an employee. It extends DB2Base 224 so the application 204 developer would not have to write the logic to access DB2 210, but could concentrate on the information about the employee. The DAPI 202 developer would need to know about the employee table (table and field names, for example) and would implement methods for selecting and displaying data. In an exemplary embodiment, DAPIUserProfile class methods include the following:

260 selectEmployeeByEmpID(employeeID,companyCode,
countryCode)

261 selectEmployeeByWebID(employeeWebID)

262 selectEmployeeBothWays(employeeID,companyCode,
countryCode,employeeWebID)

263 selectEmployeeByName(lastName,firstName)

264 insertEmployee(columnNames,DB2ColumnValues)

265 updateEmployeeByEmpID(employeeID,
employeeCompanyCode,employeeCountryCode,
UpdateNameValues...)

266 updateEmployeeByWebID(employeeWebID,
UpdateNameValues,UpdateByUserID)

267 deleteEmployeeByEmpID(employeeID,companyCode,

```

countryCode,UpdateByUserID)
268 deleteEmployeeByWebID(employeeWebID,
UpdateByUserID)
269 deleteEmployeesWhere(Condition, UpdateByUserID)
5 270 clearTable()

```

Once the data is selected, a few of the properties that are available for a given employee include empWebID, empLastName, empFirstName, empIntPhoneNum, empExtPhoneNum, empEmailID, empID, empCompanyCode, empCountryCode, and empCountryName.

An application 204 like the human resources (HR) application would then need to read HR data and insert it into the employee table if the employee did not exist, or update it if something changed, or delete it if the employee no longer exists. This application developer would then only have to know the methods and properties of the userProfile class 226 in order to write the application. An example of such an application is set forth in Table 2, with reference to the steps of Figure 14. This table sets forth the HR load routine, a batch program to read HR data from a flat file and insert it into the DB2 user profile table.

TABLE 2 EXAMPLE APPLICATION

```

1      'in the following step, instantiate a DB2 config
2      object, and call it db2; the database name 230, etc.,
3      is determined by instantiating the db2 config object,
4      as defined by the core programmers'

5      272: Dim db2 As New DB2Config(session)

6      'the database information is known, and can be passed
7      to the employee profile.'

8      274: Print "The target DB2 database alias is "&
9            db2.getDB2DatabaseName()

10     'All that must be done is to pass the DB2 class to the
11     userProfile.'

12     276: Dim eps As New userProfile(db2)

13     'Delete everything from the employee table to start the
14     bridge.'

15     278: I headerDivision = "" Then
16           Call eps.clearTable()
17     Else
18           Call eps.deleteEmployeesWhere("COGRP_CD= ""
19           &headerDivision &""")
20     End If

21     280: For count = 2 To records-1

22           'Read the next record and make sure that it can be
23           loaded without problems'

24           If ReadInputFile(inputFileNum, count, userid, al, cl
25           eps, cci) Then

26               Call eps.insertEmployee(DB2ColumnNames,
27               BuildDB2ColumnValues())
28               db2kAdditions = db2Additions+1

29           End If

30     Next...
```

In Table 3, a pseudo code example of use of the webquerysaveagent process is illustrated.

TABLE 3 WEQUERYSAVEAGENT EXAMPLE

002050"554550
1 Dimension db2 As New DB2Config(session)
2 Print "The target DB2 database alias is "&
3 db2.getDB2DatabaseName()
4 Dimension eps As New userProfile(db2)
5 execute process 260 to selectemployee by employee id
6 if employee does not exist, then
7 execute process 264 to insert employee
8 else if employee changed, then
9 execute process 265 to update employee
10 else (employee not changed) information to user
11 "employee not changed"
12 no save
13 endif

In the example of Table 3, an application programmer 204 is using a dapi written by programmers 202.

In this manner, the relational database 210 is used as

the data source, instead of a Notes database, in a way that
hides the complexities of DB2 database programming. That
is, in a fashion to similar Notes programming - the idea is
to allow a Notes programmer to use a familiar looking class
5 226 to load and save data.

4. *Catalog Administration*

In accordance with the preferred embodiment of the
invention, a requisition catalog administration function
provides control, audit, and publishing procedures for flat
10 files received from suppliers.

Referring to Figure 15, a system architecture for
implementing catalog administration includes a requester
browser 410, a buyer browser 412, with net.data connections
391 and 393 to a dedicated DB2 server and DB2 database 390
15 having a staging table 392 and a production table 394
through network dispatcher 102 and Go cluster 104. Go
cluster 104 is also connected through network dispatcher 106
and Domino cluster 112 to Domino application server 114.
A buyer 412 accesses staging table 392 via net.data
20 connection 391, and a requestor 410 accesses the production

394 table via net.data connection 393. This connection 391,
393 is implemented as a single path, and the requester and
buyer provided different levels of authority to access
different tables 392, 394 in DB2 390 over that same path.

5 Buyer 412 can change selected fields in the staging table
392 and can update production table 394 from staging table
392. Requester 410 can only view (not change) the
production table 394. The buyer at browser 412 is
controlled by a GUI which contains access control list (ACL)
10 control on fields, and edit authority for catalog access.

Referring to Figure 16, this architecture further
includes a catalog flat file 314, an application program 384
within application server 114, catalog administration
function 386, Req/Cat Web function 388, and WEB
15 communications 396 and 398 connecting a catalog
administration function with ACL control 400 and requester
402 to database 390.

In operation, catalog flat file 314 is received by
application server 114 through firewall 380 via EDI and
20 loaded into DB2 database 390 by application program 384.
Catalog administration function 386 specific users 400 audit
control over certain fields in staging table 392, and
publishes the catalog data to the live, or production,

system 394. Function 386 presents to buyer 400 a staging table 392 with a GUI front end, with selected fields enable and other fields not enabled to be personalized.

Catalog file 314 is a flat file containing catalog
5 items in a column delimited format specified to supplier 300 by the enterprise.

Application server 114 manages database 390 containing staging table 392 and production table 394. A catalog file 314 comes to application server 114, which includes a
10 program 384 for moving data from that flat file to staging table 392.

A buyer at terminal 400 accesses the staging table 392 on the web 396. He views catalog items and enters transactions with action button which transfers information
15 from staging table 392 to production table 394. Production table 394 is referenced by req cat web 388, and staging table 392 is referenced by the catalog administration function 386 operated by the buyer 400. Typically, a buyer is member of procurement organization with responsibility
20 for negotiating deals with suppliers. A requester 402 accesses production table 394 over web 398 to create and submit a requisition to SAP 382.

In accordance with the preferred embodiment of the invention, control over what buyer 400 can change is provided by a GUI in a process which loads a catalog 392 from a supplier into a production system 394.

5 Catalog files 314 come in from suppliers in an enterprise defined standard format, and the access to fields in that format is hard coded into the catalog administration function 386. Application server program 384 has error checking functions to assure validity of a catalog 314 from
10 a supplier 300. Buyer 400 accesses staging table 392 through a catalog administration function 386 which has hard coded into it the access controls on the various fields in the catalog format. Production table 394, which is accessed by the requester 402, is updated periodically (upon buyer
15 actuation) from the staging table 392.

Implementation of the invention involves several code procedures: there is a program 384 which loads a file 314 that is received via EDI into a table 392 in DB2. There are routines 388 which allow a buyer 400 to browse certain
20 catalogs in the staging table 392 and change certain fields while being inhibited from changing others. And there are the routines 386 which take the approved catalog and migrate the data from the staging DB2 table 392 to the production

DB2 table 394.

Referring to Figure 17, a preferred embodiment of these processes are presented. In supplier system 300, supplier source data 310 is extracted and reformatted in step 312 to create catalog flat file 314 in the format specified by the enterprise. In step 316 that flat file is transmitted to the enterprise 302, as is represented by line 305, where it is accepted in step 320 into the enterprise EDI mailbox 322. In step 324, the data in the flat file in mailbox 322 is reformatted and put into generation data group (GDG) 328, a location for saving more than one file, so as to retain the last N iterations, and a archive entry made to processing log 326. In step 330, a delivery component executes to send data from GDG 328 to application server 114, as is represented by line 303, in the form of catalog flat file 340. In step 342, a delivery component receives the flat file and, as is represented by line 347, starts job scripts including MASSLOAD for reading the flat file and loading staging table 392, and as represented by line 345 alerts the buyer 352. As is represented by lines 311, 313 and 315, respectively, MASSLOAD 344 accesses database server 306 procedures catalog_s 360, product_s 362, and Req/Cat Web 364, and makes an archival entry to processing log 346.

procedure 350, then buyer 352 intervention is required via GUI 370.) After procedure 364 completes execution, it may

either stop, or if catalog 366 has a flag set on and catalog_s staging table 360 indicates no critical errors,

5 then procedure 364 will invoke validation and migration procedure 350. After validate procedure 364 completes, it has written to prod_message_s file 332, and the buyer may use GUI 370 to read messages from file 332 and make any desired changes to staging table 362. They buyer may also

10 choose to reject the catalog and, via step 354, contact the supplier to restart the process. This occurs if there is an error in the unit prices, which is an example of information in the catalog which a buyer is not authorized to change on his own.

15 After the buyer has used GUI 370 to make the values in staging table 362 acceptable, he sets the flag in staging table 360 which allows migration procedure 350 to run to move data from staging table 362 into production table 368, a relational database, such as Net Commerce (NC) or IBM DB2.

20 5. Role Table GUI

A preferred embodiment of the invention provides through use of a role table in DB2 database 129 (Figure 3, or 390 in Figure 16) very flexible access to DB2 tables without requiring involvement by a database administrator (DBA) to issue grants against the tables, thus bypassing the problem caused by Notes agents all coming from the same user (the Notes server ID).

Everything in Lotus Notes, even code, is in documents which require access control list (ACL) controls on access. Consequently, the preferred embodiment of the invention uses Notes ACLs to access code. However, when accessing data, a role table 420 (see Figure 19) is used to build roles and permissions, and an object model is provided to generically access data from database 210, thus extending Notes to access a non-Notes data source 210. In order to configure DB2 to work in a Notes application environment, a single sign off is provided after getting through Notes code ACLs. This does not involve use of any of DB2's role tables and grants, but rather a single web ID 434 known to the Notes code to access the DB2 data.

Referring to Figure 19, role table 420 includes for each of a plurality of user WEB ID's 422, the associated role 424 and level 426 of granularity at which the user is

associated with the role. Example: for a role 424 of country administrator, the level 426 is the country id, and user with web ID 422 of 02 can update contract profiles for that country.

5 Any person at a browser 100 attempting to access a row in a DB2 table 390 must pass the role table 420 check. Further, for accessing a supplier table in DB2 390, anyone can view the list of suppliers in the application that applies to the requesters country, but only the country administrator can update them.

10 User 422 identifies a user profile 430, which specifies the user name 432, web identifier 434, charge information 436, including country, company, work location) and delivery information 438 (including street, office, and building).

15 Thus, in accordance with the preferred embodiment of the invention, the web ID 434 for the browser user is used to control access to the Notes databases and to the DB2 databases. The Notes databases have code and documents which the user must access (including contract profiles, cost center profiles) and also tables in the DB2 database. A user must access both Notes databases and DB2 databases, and access to all of these databases is controlled based on

the user web ID 434 through the use of role table 420.

In order to make and use the preferred embodiment of the invention, an implementer and user would do the following:

- 5 1. Determine what levels of granularity are relevant to the application. (For example, the company that the user belongs to, the country, etc.)
2. Populate the DB2 table 420 with Web IDs 434 and associated roles 424 and levels 426.
- 10 3. Write procedures to locate a user 422 in table 420 and pull out associated roles 424 and levels 426.
4. Provide code routines or functions using these procedures which are authorized for execution by users with specific roles and levels.

15 Code using these routines would then compare the roles and levels to the specific role and level that is required based on the function. For example, a user could be defined as a country administrator for France and a company administrator for a small company in the US. A routine that updates

accounting information for the small company would not care about the country-level authority, so would look in role table 420 for company administrator role 424 for this user web ID 422. The level 426 of the role 424 would further
5 restrict this user 422 from updating the accounting information for any company in the US other than the one corresponding to the level 426 to which he is assigned.

The invention allows a person's access to DB2 tables to be limited by the contents of a second db2 table rather than
10 the grants issued by the DBA. In the Notes environment of the preferred embodiment of the invention, the ID which is actually granted the authority to the table is the Notes Server machine since the server accessing the db2 tables is the Notes server. Since the user web id is once removed,
15 this provides a mechanism for applying a level of authority to the user to then apply to the db2 table. That mechanism is the role table.

Table 4 lists and describes the Req/Cat Web tables of the preferred embodiment of the invention.

20 **TABLE 4 REC/CAT WEB ADMINISTRATION TABLES**

004050"5674553

| ADMINISTRATION FUNCTION | TABLE | DESCRIPTION |
|-------------------------|------------------------|--|
| | REQCAT.TCATLG_APPROVAL | Allows a catalog administrator to flag a catalog item as requiring chemical or safety approval in specific locations. For example, toner is not considered a chemical item except in Vermont, because of special environmental laws in that state. |
| Buyer routing | REQCAT.TBUYER_PROFILE | Defines the owner of a buyer code and contact information |
| | REQCAT.TBUYER_ROUTING | Links a buyer to a commodity |
| | REQCAT.TBUYER_SUPPLIER | Links a buyer to a supplier |
| Catalog profiles | RC.CATALOG | Defines the characteristics of a catalog - supplier, expiration date, currency, etc. |
| | RC.CATPLREL | Defines the plants which are allowed to access this catalog |
| Commodity families | NC.CATEGORY | Defines the global list of commodity groupings |
| Commodity codes | RC.COMMOCODE | Defines the global list of commodities and identifies which family each belongs to |
| Company commodities | RC.COMMCOMP | Not all commodities may be valid for all companies. This table identifies which commodity codes the company wants to use. |
| | REQCAT.TCOMMCOMP_BUYER | Some commodities require that the user select a buyer from a predefined list. This is the predefined list. |
| | REQCAT.TCOMM_COMP_COA | This associates GL account codes with the commodity code. |

| ADMINISTRATION FUNCTION | TABLE | DESCRIPTION |
|-------------------------|--------------------------|--|
| | REQCAT.TCOMM_WLOC_RCV | This defines for which work locations this commodity is 'receivable'. This flag is forwarded to SAP for further use in receiving locations |
| | RC.SUBCOMMODITY | Some commodities are too broad and the need exists for sub-dividing the goods under this commodity so that different suppliers and different purchase processes can be used. |
| | RC.SUPPSUBCOMM | This links a supplier to a specific subcommodity. |
| Companies | RC.COMPANY | Associates SAP company codes with associated country. For example, IBM US contains three company codes for IBM, Lotus, and Tivoli. |
| Countries | RC.COUNTRY | Holds the list of ISO country codes, i.e., US, FR, DE, etc |
| Company groups | REQCAT.TCOMPGRP_TYP_PROF | Defines the list of valid grouping types, such as ACCOUNT, UPROF |
| | REQCAT.TCOMPGRP_PROFILE | Defines the list of group names and links them to their types, such as IBMUS - ACCOUNT and IBMUS - UPROF |
| | REQCAT.TCOMPANY_GROUPING | Lists the company codes that belong to the specified grouping |
| Plants | RC.PLANT | Associates plant codes with company and country. A company can have many plants, a plant may belong to only one company. |
| Suppliers | RC.SUPPLIER | Defines the characteristics of a supplier - name, code, contact information, location |

| ADMINISTRATION FUNCTION | TABLE | DESCRIPTION |
|-------------------------|---|--|
| | RC.SUPPCOMP | Defines which company codes may reference this supplier for purchasing |
| Work locations | REQCAT.TWORK_LOCATION REQCAT.TLWORK_LOCATION | Associates work locations with plant, company, and country. A plant can have one or more work locations, a work location may belong to only one plant. |
| | REQCAT.TWORKLOC_DELTOADDR | For those work locations which have a predefined CDC (Chemical Delivery Center) address |
| User profiles | REQCAT.EMPLOYEE_PROFILE | Holds employee information |
| Catalog parts | see Figure 18, DB2 NCF tables 368 | Hold part information, category / subcategory information, etc. |

In accordance with a preferred embodiment of the invention, in a Notes/DB2 hybrid environment, a Notes agent reads data from a DB2 table, and then dynamically populates that data to an HTML page. In this manner, an the existing infrastructure (including Notes, Domino Go, and DB2) is used to combine HTML and Java script web presentation with DB2 data.

display the web page differently depending thereon. A dynamic feature of the invention is that straight text may be displayed, or with text with hyperlinks to open a supplier profile, as an example.

5 To make and use this preferred embodiment of the invention, the following is done:

1. Use the Lotus Script lsx:lc connector connect to DB2.
2. Depending on the DB2 table being read and the functions required, write functions to Create, Read, Update, and
10 Delete with respect to the DB2 table.
3. Write the HTML to display the page, and then have the Notes agent 440 Print these HTML commands to the browser so that they appear in a meaningful presentation to the end user.
- 15 4. Use conditional logic to change the look of the page 442 based on the results of the DB2 390 lookups.

Inputs to the method of this embodiment of the invention include the DB2 table to be read or updated, and the output includes HTML conditionally generated based on

results of the DB2 table reads.

The HTML page being populated may, for example, provide a list of suppliers. Such pages may also be used in the requisition or configuration area of the application, displaying commodity codes, suppliers, and so forth.

TABLE 5 EXAMPLE AGENT 440

1 This agent is invoked from the administration tab for 'supplier'. It is
2 presenting the user with a view of suppliers that have been configured
3 on the system. Based on the access authority of the user, this list
4 will be presented either as just text, for the general user, or as
5 hotlinks for an administrator to then open an individual supplier
6 profile and update it.

7 Sub Initialize
8 //setting up variables
9 Dim src As New LCConnection ("db2")
10 Dim fldLst As New LCFieldList(100)
11 Dim suppname As LCField
12 Dim suppcode As LCField
13 Dim InfoView As AllInfoView
14 Dim session As New NotesSession
15 Dim doc As NotesDocument
16 Dim sqlQueryString As String
17 Dim flag As Variant
18 Dim admin As Integer
19 Admin=True
20 Dim lclsxSession As LCSession
21 Dim supplierdb As notesdatabase
22 Dim configview As notesview
23 Dim configdoc As notesdocument
24 Dim lookuptype As String
25
26 On Error Goto errHandler
27
28 Set supplierdb=session.currentdatabase
29 Set configview = supplierdb.getview("APPVIEW")
30 Set configdoc = configview.getfirstdocument
31 lookuptype = configdoc.HRFormat(0)
32
33 Set doc=session.DocumentContext
34 Set InfoView=New AllInfoView
35
36 //connect to DB2


```

86      './SUPPLIER?OpenForm';"" >"
87          Print "</form>"
88      Else
89          Print "<BR>"
90      End If
91
92      Print "<table cellpadding=2 cellspacing=0 border=0>"
93      Print "<tr><td class=""bannersublevel"" ><B> Supplier</B></td>"
94      Print "</tr>"
95
96      //Toggling background color on alternate rows
97
98      flag = True
99
100     While (src.Fetch (fldLst) > 0)
101         If (flag) Then
102             Print "<tr bgcolor=#CCCCC>"
103             flag = False
104         Else
105             Print "<tr bgcolor=#FFFFFF>"
106             flag = True
107         End If
108
109         //If user is an admin, then print the supplier name as a hotlink
110
111         If Admin Then
112             Print"<td class=""field""> <a href=./SUPPLIER?OpenForm&"
113             & suppcode.text(0) & ">"&suppname.text(0)&"</a> </td>"
114             Else
115
116             //otherwise just print it as text
117
118             Print"<td class=""field"">" &suppname.text(0)&"</td>"
119             End If
120             Print"<td class=""field"">" &suppcode.text(0)&"</td>"
121             Print "</tr>"
122         Wend
123         Print "</table>"
124         End
125
126     errHandler:
127
128     Print "<br>" & session.currentagent.name & " - Line # " &Str(Erl)
129     & " // Error " & Str(Err) & ": " & Error$
130     If (lclsxSession.Status <> LCSUCCESS) Then
131         Dim text As String
132         Dim extcode As Long
133         Dim exttext As String
134
135         Call lclsxSession.GetStatus (text, extcode, exttext)
136         If (lclsxSession.Status = LCFAIL_EXTERNAL)Then
137             Print "<br>DB2 message: " & exttext & " code #" &
138             Cstr(extcode)
139         Else
140             Print "<br>Connector message: " & text
141         End If
142     Else

```

139 Print Error\$
140 End If
141
142
143 End Sub

7. Data Transfer

5 In accordance with the preferred embodiment of the invention, a system and method is provided for transferring to a Req/Cat Web (RCW) application in a timely manner large quantities of data from web sites that exist outside of a firewall, or internal applications within the firewall but outside of the Req/Cat Web application.

10 For the purpose of the description of this embodiment, RCW exists in a frame set in a browser. A frame set divides a screen into logical and user-friendly sections called frames. As part of security for browser 100, Netscape and Internet Explorer (IE) establish ownership of the frames: each frame is a window under control of browser 100. Browser 100 can open up a session in any frame desired.
15 However, if all of the frames open on a browser are not owned by same session, then these frames cannot see the contents of each other.

Consequently, the problem is presented: if RCW needs to

access an outside supplier site 300 for information to get
back through one its own the frames, as soon as data is
written by that other site into one of the frame at browser
100, RCW no longer owns the frame and cannot access the
5 information. So this aspect of the invention is concerned
with data passing.

In accordance with the preferred embodiment of the
invention, the supplier 300 opens a new window through
normal Java script commands and writes its data into it,
10 along with the name of the agent to run in RCW application
server 114. That new window then calls one of the RCW
processes, which is able to see the data because it is not
in a frame and is therefore available for RCW to access.
That process causes an agent to runs (in RCW) that can see
15 that data and write it to a frame on the RCW side, resulting
in RCW owning both the data and the frame. This allows RCW
to reach NOTES information, and write that information to
other frames. RCW owns the data and the frame. Ownership
is established in the RCW application.

20 Consequently, by carefully tracking and controlling the
sequence of opening windows, RCW can establish the origin
from which data is coming, where the data is going, and
which process owns it.

Those are the high points. Review: two sites are talking to each other using an intermediate window, doing a handoff of data and of ownership of both data and frame such that when it is time to supply information to the RCW application, RCW owns the data, frame and process.

Referring to Figure 21, a system and method are provided for data transfer from a externally owned site to an application owned frame set which operates as follows. The RCW application opens a supplier site URL in a temporary frame 456. Upon user selection of go to supplier 451, as is represented by step 458, frame 456 opens the supplier window 460 as a separate browser session. Two windows are now open: the original application 450 with its window underlying, and a supplier window 460 over it with the supplier URL. The reason for doing this is that the supplier requires that the browser be full frame, not in a small frame set. The primary RCW application in window 450 is quiesced to a wait state. As is represented by steps 462, the user can now select from window 460 items to buy, search, or whatever the supplier deems is appropriate for a user to order his data. The user then issues the command to submit the order. In step 444, the supplier site then gathers content from order data entered at window 460, in step 446 formats the page, and in step 448 issues a call to

yet another Notes database.

The key consideration here is, "who owns the frame".
The server that owns the agent that last wrote a frame owns
it, and can see its content, but cannot see any other
5 frame's contents if they were written by another server's
programs or agents.

Window 464 and first agent 480 act as an intermediary.
Content 466 is owned by the supplier server 300, but the
first agent 480 is owned by the Req/Cat Web server 114.
10 That first agent 480 then kicks off the rest of the process,
passing the data and ownership to the RCW server 114 and the
second agent 484. The second agent 484 can now can access
Notes data on the Notes server 112, and issue the commands
that send the order to be stored in the requisition.

15

8. Customizable Side Bar

In accordance with the preferred embodiment of the
invention, a customizable side bar is provided. Dynamic
HTML is used by the navigator responsive to user input to
change data presented at the screen without having to

communicate with a remote server.

Referring to Figure 4, navigation frame or menu bar appears, typically, on the left of a display window, to display a plurality of menu boxes, including headers 491-494 and items 495-500. As a cursor is moved over the headers, each individual header is highlighted or some symbol 481, 483, 485, 487, respectively, rotated by, say, 45 degrees, so as to point either down or to the right, to indicate to the user the header which will, upon being clicked, toggled to either a collapsed or expanded state from its current expanded or collapsed state, respectively. As illustrated, headers 481 and 483 are in a collapsed state, and headers 485 and 487 are in an expanded state. When expanded, header 485 is expanded to show items 495-498, and header 487 to show items 499-500. The user may move the cursor to one of items 495-500, and select the item to update the data displayed in content frame 480.

Heretofore, when the user selects (clicks on) a menu header 491, there is generally a pause as the request is made to the server to obtain a new page including an expanded menu bar 472 including a display of the included items. In accordance with a preferred embodiment of the invention, when server loads the window to a client, all of

the information required to load the menu bar 472 is provided, including information for the expanded menu items (but not the content frame associated with them). If a menu header 485 is clicked when in the collapsed state, the menu items 496-498 are shown or made visible and the following menu headers and items are moved relative to the expanded menu bar. If a menu item 496 is clicked, then communication with the server is required to load the content frame 480.

Upon initial load, all headers 491-494 are collapsed and all menu items are hidden. Menu headers 491-494 are displayed below each other with no gaps between them. Clicking on a last menu header merely shows all the menu items and moves nothing below it. Clicking on the first menu header would show all of its items and move every header and item a constant amount without changing its visibility state. (The display area 472 may be off of the window, but is available via the scroll bar 490.) In accordance with this embodiment of the invention, the result of selecting a header 491-494 is an instantaneous response from the client browser either expanding or collapsing the selected header and respectively displaying or removing from display the included items.

This is done using dynamic html (dhtml), which allows

creation of divisions within a document. These divisions are equivalent to tab items, menu items 495-500, or headers 491-494. In Netscape, these divisions are called layers. In Explorer, the layers are called divisions. These divisions can be hidden or shown, and moved relative distances on the screen. The current state of the art is to use these divisions for moving or flashing graphics, but not for business applications such as menu bars. In accordance with the present invention dhtml at the client creates subtle changes, business like, in the menu bar 472 without requiring server communications. The use of dhtml is described at developer.netscape.com.

Java script code, executable at a client for inserting, showing, moving and updating a menu bar 472, is illustrated in Table 6.

TABLE 6 MENU BAR CODE

```
1  ---- JavaScript (appNav.js) -----
2  // appNav Class Constructor
3  -----
4  // This class implements a JavaScript Object intended to represent the
5  Navigator function  appNav(menuVar)  {
6
7      //methods
8      this.init = appNavInit;
9      this.sizeit = appNavSizeit;
10     this.toggletext = appNavToggleText;
11     this.isItem = appNavIsItem;
12     this.reverse = appNavReverse;
13
14     // properties
```

```

13      this.ns = document.layers;
14      this.ie = document.all;

15      this.loaded = 0;
16      this.whichone = 0;
17      this.whichgroup = 0;
18      if ( this.ns ) {
19          this.show = 'show';
20          this.hide = 'hide';
21      } else {
22          this.show = 'visible';
23          this.hide = 'hidden';
24      }
25      this.menus = menuVar;
26      this.max = menuVar.length;
27      this.images = new Array(this.max);
28      this.menuMove = new Array(this.max);
29      for (i=0;i < this.max; i++) this.menuMove[i] = 20 * (menuVar[i] -
30 1);
31      this.tabShow = [false, false, false, false, false, false, false,
32 false];
33  }

34  function appNavInit() {
35      var k=0;
36      if (this.loaded == 0) {
37          for (i=0; i < this.menuMove.length; i++) {
38              this.images[i] = new Array(this.menus[i]);
39              for (j=0; j < this.menus[i]; j++) {
40                  this.images[i][j] = new Image();
41                  pos = ("0"+i).slice(i>9,2) + ("0"+j).slice(j>9);
42                  this.images[i][j].src = "../images/men" + pos
43  + ".gif"
44                  if (this.ie) {
45
46 document.images[k].src=this.images[i][j].src;
47                     k++;
48                 } else
49
50 document.layers["D"+pos].document.images[0].src=this.images[i][j].src;
51                 }
52             }
53             this.loaded=1;
54         }
55     };

56  function appNavSizeit() {
57      if (this.loaded==1) {this.loaded = 0; this.init();}
58  };

59  function appNavToggleText(z) {
60      if (this.loaded==1) {
61          this.tabShow[z] = !this.tabShow[z]
62          for (j=1; j<this.menus[z]; j++) {
63              pos = ("0"+z).slice(z>9) + ("0"+j).slice(j>9);
64              if (this.ie)
65                  text = document.all("D"+pos).style

```

```

66         else
67             text = document.layers["D"+pos]
68         if (this.tabShow[z])
69             text.visibility = this.show;
70         else
71             text.visibility = this.hide;
72     }
73     for (i=z+1; i<this.max; i++) {
74         for (j=0; j<this.menus[i]; j++) {
75             pos = ("0"+i).slice(i>9) + ("0"+j).slice(j>9);
76             if (this.ie) {
77                 text = document.all("D"+pos).style
78                 if (this.tabShow[z])
79                     text.pixelTop += this.menuMove[z]
80                 else
81                     text.pixelTop -= this.menuMove[z]
82             } else {
83                 text = document.layers["D"+pos]
84                 if (this.tabShow[z])
85                     text.top += this.menuMove[z]
86                 else
87                     text.top -= this.menuMove[z]
88             }
89         }
90     }
91 }
92 };

93 //Now we will need some DHTML to run this javascript on...

94 <html><head>
95 <title>Requisition Catalog Navigator</title>
96 <script src="appNav.js"></script>
97 <script language="JavaScript">
98     menus = [7, 6, 4, 3, 22, 1, 1, 1];
99     myNav = new appNav(menus);
100 </script>
101 <style type="text/CSS">
102     div { position:absolute; color:black; visibility:hide;
103     visibility:hidden;
104     border-width:0px; width:125px; height:20px; border:0px;}
105 </style>
106 </head>

107 <body bgcolor="#000000" onresize="myNav.sizeit()" onload="myNav.init()">

108 <div ID=D0000 style="top:0px; visibility:show; visibility:visible;">
109     <a href="#" onClick="myNav.toggletext(0); return false;">
110         <img alt="Create Requisitions" width=125 height=40
111         border=0></a></div>
112 <div ID=D0001 style="top:40px;">
113     <a
114     href="javascript:top.jsCode.validateInitiator(top.DBCreateReq+'WebReq?O
115     penForm&NEW',top.reqData)">
116         <img alt="New" width=125 height=20 border=0></a></div>
117 <div ID=D0002 style="top:60px;">
118     <a

```



```

233      <a
234 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WBUSINESSENTITIES?Open
235 View')">
236      <img alt="Business Units" width=125 height=20 border=0></a></div>
237 <div ID=D0406 style="top:260px;">
238      <a
239 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WCAPITALAPPROVERS?Open
240 View')">
241      <img alt="Capital Approvers" width=125 height=20
242 border=0></a></div>
243 <div ID=D0407 style="top:280px;">
244      <a
245 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/AgentRunner?OpenForm&V
246 iewCatalog&')">
247      <img alt="Catalogs" width=125 height=20 border=0></a></div>
248 <div ID=D0408 style="top:300px;">
249      <a
250 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WCHEMICALAPPROVERS?Ope
251 nView')">
252      <img alt="Chemical Approvers" width=125 height=20
253 border=0></a></div>
254 <div ID=D0409 style="top:320px;">
255      <a
256 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/AgentRunner?OpenForm&V
257 iewCommCode&')">
258      <img alt="Commodity Codes" width=125 height=20 border=0></a></div>
259 <div ID=D0410 style="top:340px;">
260      <a
261 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/AgentRunner?OpenForm&V
262 iewCommFam')">
263      <img alt="Commodity Families" width=125 height=20
264 border=0></a></div>
265 <div ID=D0411 style="top:360px;">
266      <a
267 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WCOMPANIES?OpenView') "
268 >
269      <img alt="Companies" width=125 height=20 border=0></a></div>
270 <div ID=D0412 style="top:380px;">
271      <a
272 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WCOMPCOMMODS?OpenView'
273 )">
274      <img alt="Company Commodities" width=125 height=20
275 border=0></a></div>
276 <div ID=D0413 style="top:400px;">
277      <a
278 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WCONTRACTS?OpenView') "
279 >
280      <img alt="Contracts" width=125 height=20 border=0></a></div>
281 <div ID=D0414 style="top:420px;">
282      <a
283 href="Javascript:top.jsCode.tocontent(top.DBCostCtr+'/WCOSTCENTERS?OpenV
284 iew')">
285      <img alt="Cost Centers" width=125 height=20 border=0></a></div>
286 <div ID=D0415 style="top:440px;">
287      <a
288 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WCOUNTRIES?OpenView') "
289 >

```

```

290         <img alt="Countries" width=125 height=20 border=0></a></div>
291 <div ID=D0416 style="top:460px;">
292     <a
293 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WFINANCIALAPPROVERS?Op
294 enView')">
295         <img alt="Financial Approvers" width=125 height=20
296 border=0></a></div>
297 <div ID=D0417 style="top:480px;">
298     <a
299 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WPLANTS?OpenView')">
300         <img alt="Plants" width=125 height=20 border=0></a></div>
301 <div ID=D0418 style="top:500px;">
302     <a
303 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WSAFETYAPPROVERS?OpenV
304 iew')">
305         <img alt="Safety Approvers" width=125 height=20
306 border=0></a></div>
307 <div ID=D0419 style="top:520px;">
308     <a
309 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/AgentRunner?OpenForm&V
310 iewSupplier')">
311         <img alt="Suppliers" width=125 height=20 border=0></a></div>
312 <div ID=D0420 style="top:540px;">
313     <a
314 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WTaxLocs?OpenView')">
315         <img alt="Tax Locations" width=125 height=20 border=0></a></div>
316 <div ID=D0421 style="top:560px;">
317     <a
318 href="Javascript:top.jsCode.tocontent(top.DBCfg+'/WORKLOCS?OpenView')">
319         <img alt="Work Locations" width=125 height=20 border=0></a></div>
320 <div ID=D0500 style="top:160px; visibility:show; visibility:visible;">
321     <a href="/transform/reqcat/confirm/rccnfirm.nsf?Open" target=_top>
322         <img alt="Go To Confirmations" width=125 height=20
323 border=0></a></div>
324 <div ID=D0600 style="top:180px; visibility:show; visibility:visible;">
325     <a href=# onClick="openCfg('/GoToCatAdmin?OpenAgent');">
326         <img alt="Catalog Administration" width=125 height=20
327 border=0></a></div>
328 <div ID=D0700 style="top:200px; visibility:show; visibility:visible;">
329     <a href=# onClick="openHelp();">
330         <img alt="Req/Cat Web Help" width=125 height=20
331 border=0></a></div>
332 <div ID=D0800 style="top:940px;"></div>
333 </body></html>

```

Every division that ends in 00 is a header (Dxx00) all the other divisions with the same header number are items under that division. (D 01 00 - is the header for the second group of items) (D 01 01 - D 01 06 are the six items

5 under the second group). Netscape recognizes the div tags
as layers. Internet Explorer recognizes them as layers.

Advantages over the Prior Art

05637495 "05637495"
10 It is an advantage of the invention that there is
provided a scalable database system in a web environment
with optimal access performance characteristics for an
expanding number of clients and a growing database.

15 It is an advantage of the invention that there is
provided a system and method for enabling a requester to
select the correct general ledger account when doing
financial validation on a requisition.

20 It is an advantage of the invention that there is
provided a system and method for utilizing Lotus script
extensions in combination with a relational database to
provide high capacity storage without performance
degradation.

It is an advantage of the invention that there is provided a system and method for enabling a buyer a means for editing catalog content before externalizing it to production for access by requesters.

25 It is an advantage of the invention that there is provided a system and method for providing very flexible access to DB2 tables without requiring database administrator (DBA) involvement to issue grants against the tables, and bypassing the problem caused by Notes agents all
30 coming from the same user (the Notes server ID).

It is an advantage of the invention that there is provided a system and method utilizing an existing infrastructure including Lotus Notes, Domino Go, and DB2 to combine HTML and Java script web presentation with DB2 data.

35 It is an advantage of the invention that there is provided a system and method for enabling transfer of data from a supplier site to a RCW application which does not entail frame spoofing.

40 It is an advantage of the invention that there is provided a system and method for dramatically increasing the speed of operation of a navigation frame of a GUI.

Alternative Embodiments

It will be appreciated that, although specific
embodiments of the invention have been described herein for
45 purposes of illustration, various modifications may be made
without departing from the spirit and scope of the
invention. In particular, it is within the scope of the
invention to provide a computer program product or program
element, or a program storage or memory device such as a
50 solid or fluid transmission medium, magnetic or optical
wire, tape or disc, or the like, for storing signals
readable by a machine, for controlling the operation of a
computer according to the method of the invention and/or to
structure its components in accordance with the system of
55 the invention.

Further, each step of the method may be executed on any
general computer, such as an IBM System 390, AS/400, PC or
the like and pursuant to one or more, or a part of one or
60 more, program elements, modules or objects generated from
any programming language, such as C++, Java, Pl/1, Fortran
or the like. And still further, each said step, or a file
or object or the like implementing each said step, may be

65

executed by special purpose hardware or a circuit module
designed for that purpose.

Accordingly, the scope of protection of this invention
is limited only by the following claims and their
equivalents.

09637495-090700